2.6 Conformance of an Application With DOL

In the following, "DOL abstract syntax" means an XMI document that conforms to the DOL metamodel. Optionally, further representations (e.g. as JSON) can be supported.

- A *parser* is DOL-conformant if it can parse the DOL textual syntax and produce the corresponding DOL abstract syntax.
- A *printer* is DOL-conformant if it can read DOL abstract syntax and produce DOL textual syntax.
- DOL-conformant software that is used to *edit, format or manage* DOL libraries **must** be capable of reading and writing DOL abstract syntax. Moreover, it **must** meet the requirements for a DOL-conformant parser if it is able to read in DOL textual input. It **must** meet the requirements of a DOL-conformant printer if it is able to generate DOL textual output. However, it is also possible that a software for DOL management will work on the abstract syntax only, delegating the reading and generation of DOL text to external parsers and/or printers.
- a *static analyzer* is DOL-conformant if it can compute the logic and the signature of an OMS according to the semantics defined in section 10. In more detail, a static analyzer can have the following capabilities:
  - *simple analysis*: static analysis of DOL excluding networks and alignments;
  - *full analysis*: static analysis of full DOL.
- a *transformation tool* is DOL-conformant if it operates on DOL (abstract) syntax and implements one (or more) language translations, logic translations, language projections and/or logic projections.
- Software that implements machine *reasoning* about OMS (e.g., theorem proving, approximation) complies with this specification if and only if it interprets DOL documents according to the semantics defined in section 10. In more detail, a reasoning tool can have the following capabilities:
  - *simple logical consequence*, i.e. checking whether all sentences that are marked as `%implied` within basic OMS and extensions are logical consequences of the enclosing OMS;
  - *structured logical consequence*, i.e. checking whether all sentences that are marked as `%implied` are logical consequences of the enclosing OMS and whether all entailments in a DOL document have a defined semantics;
  - *interpretation*, i.e. checking whether all interpretations in a DOL document have a defined semantics;
  - *simple refinement*, i.e. checking whether all refinements of OMS in a DOL document have a defined semantics;
  - *full refinement*, i.e. checking whether all refinements (both of OMS and networks) in a DOL document have a defined semantics;
  - *simple conservativity*, i.e. checking whether all conservativity statements in a DOL document have a defined semantics;
  - *full conservativity*, i.e. checking whether all statements about conservative, monomorphic, definitional and weakly definitional extensions in a DOL document have a defined semantics;
  - *module extraction*, i.e. the ability to compute modules (typically, a given tool will provide this only for some logics);
  - *approximation*, i.e. the ability to compute approximations (typically, a given tool will provide this only for some logics and logic projections);
  - *full* DOL *reasoning*, i.e. checking whether an DOL document has a defined semantics.

In practice, DOL-aware *applications* may also deal with documents that are not conforming with DOL according to the criteria established in clause 2.5. However, an application only *conforms* with DOL if it is capable of producing DOL-conforming documents as its output when requested.

DOL-aware applications **shall** support a fixed (possibly extensible) set of OMS languages conforming with DOL.

It is, for example, possible that a DOL-aware application only supports OWL and Common Logic. In that case, the application may process DOL documents that mix OWL and Common Logic ontologies, as well as native OWL and Common Logic documents.